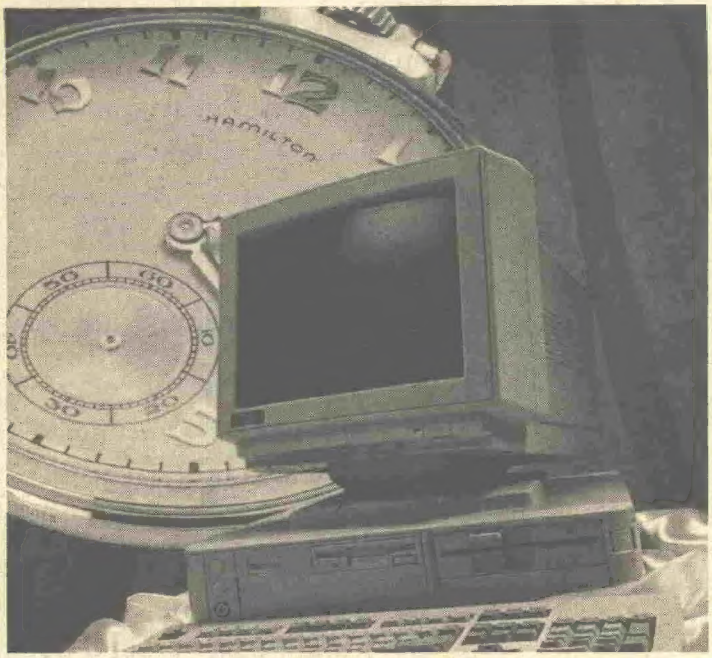
เรื่องน่าอ่าน

ดร.อัศนีย์ ก่อตระกูล

หนึ่งชั่วโมงกับ ปัญญาประดิษฐ์ ตอนที่ 2



อธิบายถึงการประดิษฐ์ความรู้ให้เครื่องคอมพิวเตอร์เพื่อความเป็นอัจฉริยะในการใช้งาน

นับตั้งแต่มีการบัญญัติคำว่า "ปัญญาประดิษฐ์" หรือ Artificial Intelligence ไว้ในทำเนียบของศาสตร์สาขาวิศวกรรม สิ่งต่าง ๆ ที่ประดิษฐ์ขึ้นใหม่ ไม่ว่าจะเป็นสิ่งที่ใช้ในชีวิตประจำวัน เช่น พัดลม หม้อหุงข้าว ตู้เย็น กล้องถ่ายรูป หรือซอฟต์แวร์ต่าง ๆ เช่น ดาต้าเบส ระบบถามตอบด้วยภาษาธรรมชาติ หรือแม้กระทั่งหุ่นยนต์มักจะมีคำว่า ... อันชาญฉลาด หรือ Intelligent รวมอยู่ด้วยเสมอ ตัวอย่างเช่น พัดลมอันชาญฉลาด Intelligent Database System, Intelligent Robot เพื่อแสดงถึงความแปลกใหม่ความทันสมัยของเทคโนโลยี ทั้งนี้ก็เพื่อความอยู่รอดของนักสร้างหรือผู้ผลิตสินค้า...เคล็ดลับสู่ความสำเร็จในการเพิ่มเงินในกระเป๋าของผู้ผลิตนั้นเกิดจากผลพวงของงานวิจัยในสาขาปัญญาประดิษฐ์

บทความ "หนึ่งชั่วโมงกับปัญญาประดิษฐ์" ที่ลงในไมโครคอมพิวเตอร์ฉบับที่ 98 เป็นเพียงภาพโดยรวมของงานทางด้านปัญญาประดิษฐ์ สำหรับวันนี้เราจะมาเจาะลึกถึงการประดิษฐ์ความรู้ซึ่งเป็นหนึ่งในเคล็ดลับของความเป็นอัจฉริยะของเครื่องจักรหรือคอมพิวเตอร์

อะไรคือ "อัจฉริยะ"

ในบรรดามนุษย์ด้วยกัน ถ้าใครได้รับฉายาว่า "มนุษย์อัจฉริยะ" ก็ย่อมหมายถึงความเป็นผู้รู้ ผู้มีไหวพริบ มีความสามารถเหนือบุคคลทั่วไป ตัวอย่างเช่น เด็กอายุ 12 ขวบ สามารถสอบเข้ามหาวิทยาลัยคณะแพทยศาสตร์ หรือเด็กหญิงอายุ 3 ขวบ สามารถปิดตาตีขิมได้ ความสามารถที่เหนือบุคคลในระดับเดียวกัน ย่อมเป็นที่ชื่นชมและได้การยอมรับว่าเก่งหรือฉลาด ทำนองเดียวกันสิ่งประดิษฐ์ต่าง ๆ ไม่ว่าจะเป็นซอฟต์แวร์หรือเครื่องจักร หากมีลักษณะพิเศษเหนือซอฟต์แวร์หรือเครื่องจักรธรรมดาย่อมเป็นที่ดึงดูดใจและได้รับการยอมรับจากผู้ใช้ ดังนั้นเพื่อความอยู่รอดในยุคของสงครามเศรษฐกิจ ผู้ผลิตสินค้าจึงจำเป็นต้องหาเทคโนโลยีที่แปลกใหม่เสริมใส่ในสินค้าของตนเอง และนี่ก็คือที่มาของทีวีระบบ AI พัดลมอันชาญฉลาด หุ่นยนต์อัจฉริยะ เครื่องซักผ้าระบบฟัซซี่ คำต่อท้ายสินค้าเหล่านี้เป็นผลพวงจากงานวิจัยทางปัญญาประดิษฐ์ซึ่งพยายามปฏิวัติให้คอมพิวเตอร์พ้นสภาพจากนักจำ นักประมวลผลที่มี

ความเร็วสูง มาเป็นคอมพิวเตอร์ที่คิดเป็น ตัดสินใจเองเป็น และรู้จักเรียนรู้

เคล็ดลับของความเป็น "อัจฉริยะ"

สิ่งสำคัญที่ทำให้มนุษย์มี "ปัญญา" ต่างจากหนูหรือสัตว์อื่น ๆ รวมทั้งมนุษย์ในยุคหินก็คือ มนุษย์สามารถพัฒนาความรู้จากการเรียนรู้ มนุษย์มีกระบวนการในการหาเหตุผลในการแก้ปัญหาและในการคิด ลักษณะพิเศษของมนุษย์เหล่านี้เป็นสิ่งที่นักวิจัยสาขาปัญญาประดิษฐ์พยายามสร้างให้กับคอมพิวเตอร์และเรียกคอมพิวเตอร์ที่มีพฤติกรรมทางปัญญาเช่นมนุษย์ว่า คอมพิวเตอร์อัจฉริยะ

เคล็ดลับในการสร้างคอมพิวเตอร์อัจฉริยะนั้นเริ่มจาก

การใช้เทคนิคฮิวริสติค

ในการแก้ปัญหาหรือคำนวณหาคำตอบ คอมพิวเตอร์สามารถทำได้ดีกว่ามนุษย์ในแง่ของความเร็ว แต่ในบางปัญหา ความเร็วของคอมพิวเตอร์ไม่สามารถค้นหาคำตอบได้ดีเท่ามนุษย์ ตัวอย่างเช่น การเล่นเกมหมากรุก ดังนั้นเพื่อให้คอมพิวเตอร์สามารถยกระดับจากการค้นหาคำตอบจากทุกเส้นทางที่เป็นไปได้ (Brute Force or Blind Search) มาเป็นการค้นหาคำตอบอย่างมีความคิด จึงมีการค้นคิดเทคนิคต่าง ๆ ที่เรียกว่า เทคนิคฮิวริสติค (Heuristic Search) เพื่อช่วยให้คอมพิวเตอร์แก้ปัญหาได้อย่างมีประสิทธิภาพมากขึ้น

การประดิษฐ์ความรู้

"ความรู้คือดวงประทีปส่องทาง" ความสามารถของมนุษย์ในการแก้ปัญหาต่าง ๆ ขึ้นอยู่กับพื้นฐานความรู้และความสามารถในการใช้ความรู้ให้เป็นประโยชน์ ตัวอย่างเช่น เรารู้ว่ามนุษย์ทุกคนต้องตาย ผู้ชายบางคนเป็นทหาร คุณภีมเป็นผู้ชาย เราสามารถสรุปได้ว่าสักวันหนึ่งคุณภีมก็ต้องตาย และคุณภีมก็น่าจะโดนเกณฑ์ไปเป็นทหาร ดังนั้นเพื่อให้คอมพิวเตอร์สามารถหาเหตุผลได้จึงมีการ "ประดิษฐ์" ความรู้ให้อยู่ในรูปที่คอมพิวเตอร์สามารถคำนวณได้ (Knowledge Representation) และ "ประดิษฐ์" กลไกในการจัดการและหาเหตุผลจากความรู้ (Knowledge Manipulation and Reasoning) รวมทั้งการสร้างภาษาโปรแกรมและเครื่องมือช่วยในการพัฒนาและจัดการกับความรู้ที่ประดิษฐ์ขึ้นมา

การเรียนรู้โดยเครื่องจักร

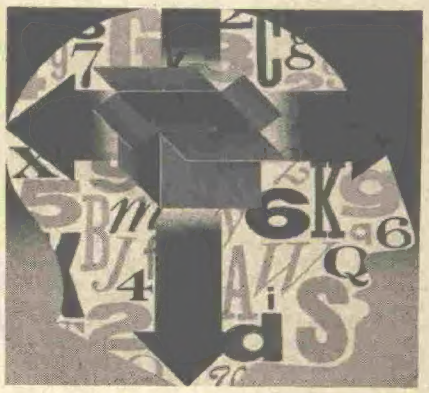
การเรียนรู้เป็นวิธีการหรือกระบวนการของมนุษย์ที่ใช้ในการเพิ่มเติมความรู้ของตนจากประสบการณ์ การเรียนรู้ของมนุษย์แบ่งออกเป็น 3 ประเภทใหญ่ ๆ คือ การเรียนรู้จากการสังเกต (Observational Learning) การเรียนรู้จากผลลัพธ์ของการแก้ปัญหา (Instrumental Conditioning Learning) และการเรียนรู้จากการฝึกฝน (Classical Conditioning Learning) ดังนั้นเพื่อให้คอมพิวเตอร์สามารถเรียนรู้ได้เช่นมนุษย์ จึงมีการจำลองลักษณะการเรียนรู้ของมนุษย์ ได้แก่ การเรียนรู้ลักษณะทั่วไป (Inductive Learning) การเรียนรู้จากการเปรียบเหมือน (Analogy Learning) การเรียนรู้จากผลลัพธ์ของการแก้ปัญหา (Case-based Reasoning) นอกจากนี้ยังได้มีการจำลองลักษณะการทำงานของสมองมนุษย์ที่สามารถประมวลผลเชิงขนานที่เรียกว่า นิวรอลเน็ตเวอร์ก หรือนิวโรคอมพิวเตอร์ (Neural Network or Neurocomputer)

"ประดิษฐ์" ความรู้ คือประตูสู่ปัญญาประดิษฐ์

การประดิษฐ์ความรู้ให้กับคอมพิวเตอร์เป็นก้าวแรกของการสร้างคอมพิวเตอร์ที่คิดได้ ตัดสินใจได้ เป็นที่ปรึกษาได้ ก่อนที่จะพูดถึงรูปแทนความรู้ที่คอมพิวเตอร์สามารถคำนวณได้ เราควรทราบถึงคุณสมบัติประจำตัวของความรู้ที่จะประดิษฐ์ขึ้น ได้แก่

การถ่ายเทคุณสมบัติ (Inheritance)

ตัวอย่างเช่น นกเป็นสัตว์มีปีก มีขน บินได้ และนกแก้วที่ชื่อจุ๋มจิ๋มเป็นนกชนิดหนึ่ง ดังนั้นความรู้ที่เกี่ยวกับคุณสมบัติของนกก็ควรที่จะถ่ายเทให้กับเจ้าจุ๋มจิ๋มได้ แต่การถ่ายเทคุณสมบัติก็มีข้อยกเว้นเหมือนกัน เช่น นกเพนกวินเป็นนกชนิดหนึ่ง แต่เพนกวินบินไม่ได้ ฉะนั้นการถ่ายเทคุณสมบัติจากกลุ่มที่มีลักษณะทั่วไปมากกว่า (Super-class ในที่นี้คือนก) ลงมายังกลุ่มที่มีลักษณะพิเศษกว่า (ในที่นี้คือ นกกระจอกกับนกเพนกวิน) จึงต้องมีทั้งค่าที่ถ่ายเทได้ (default values) กับข้อยกเว้น (exception)



ความแตกต่างระหว่างความรู้ที่มองเห็นได้จากภายนอก (Extensional Knowledge) กับความรู้ที่อยู่ในแก่นลึก (Intensional knowledge)

ตัวอย่างเช่น ความรู้เกี่ยวกับลูกบอล ทุกคนรู้ว่ากลม กลิ้งได้ กระเด้งได้ แต่ก็ไม่มีใครตอบได้ทันทีว่า กลมแค่ไหน กระเด้งได้ดีแค่ไหน กลิ้งได้ไกลแค่ไหน ความรู้ที่เกี่ยวกับภายนอกเราสามารถเก็บไว้ในฐานข้อมูล (data base) ได้ แต่ความรู้ที่อยู่ในแก่นลึกเราจะประดิษฐ์ให้กับคอมพิวเตอร์ได้อย่างไร

ความรู้เบื้องบน (meta knowledge)

ตัวอย่างเช่น อาจารย์พันธุ์ปิติเชื่อว่าอาจารย์ศิริพรรู้ว่าอาจารย์อัศนีย์สอนวิชา AI การสร้างคอมพิวเตอร์ให้มี "ปัญญา" นั้นจึงไม่เพียงแต่ทำให้คอมพิวเตอร์รู้มาก แต่ต้องให้คอมพิวเตอร์รู้จักตัวมันเองด้วยว่ารู้อะไรบ้าง มีข้อจำกัดแค่ไหน และไม่เพียงแต่แก้ปัญหาได้ แต่ต้องอธิบายได้ว่าแก้ปัญหานั้นได้อย่างไร

คุณสมบัติของความรู้ดังกล่าวจำเป็นต้องมีรูปแทนความรู้เพื่อช่วยให้คอมพิวเตอร์สามารถคำนวณได้

ดังนั้นคุณสมบัติของรูปแทนความรู้ที่ดีควรมีลักษณะดังต่อไปนี้

- แสดงให้เห็นถึงสสารวัตถุและความสัมพันธ์ของสสารวัตถุได้อย่างชัดเจน ฉะนั้นเมื่อดึงความรู้ออกมาใช้สิ่งที่เกี่ยวข้องหรือสัมพันธ์กันก็สามารถนำออกมาใช้ได้ด้วย
- สามารถถ่ายเทคุณสมบัติหรือเงื่อนไขถึงกันได้
- สามารถตัดรายละเอียดออกไปได้เมื่อไม่ต้องการใช้งาน แต่ก็ยังคงดึงออกมาใช้ได้เมื่อจำเป็น
- มีความโปร่งใส (be transparent) นั่นก็คือ เป็นรูปแทนความรู้ที่ให้ความชัดเจน เข้าใจง่าย
- มีความสมบูรณ์ (be complete) นั่นก็คือ เป็นรูปแทนความรู้ที่มีข้อมูลครบตามที่ต้องการ
- มีความรัดกุม (be concise) นั่นก็คือ สามารถแทนความรู้ได้อย่างมีประสิทธิภาพ ไม่เยิ่นเย้อ แต่ได้ใจความ
- ตอบสนองได้รวดเร็ว นั่นก็คือ เป็นรูปแทนความรู้ที่สามารถจัดเก็บหรือดึงออกมาใช้ได้อย่างรวดเร็ว
- สามารถคำนวณได้ (be computable) นั่นก็คือ สามารถสร้างรูปแทนความรู้ด้วยขั้นตอนการคำนวณหรืออัลกอริทึมที่มีอยู่

ด้วยคุณสมบัติข้างต้น จะทำให้คอมพิวเตอร์สามารถจัดการความรู้ได้โดยอาศัยกระบวนการเข้าถึง (access procedures) และกระบวนการหาเหตุผลเพื่อสร้างความรู้ขึ้นใหม่ ปรับปรุงความรู้หรือดึงความรู้ออกมาใช้

รูปแทนความรู้ที่สำคัญ ๆ ได้แก่

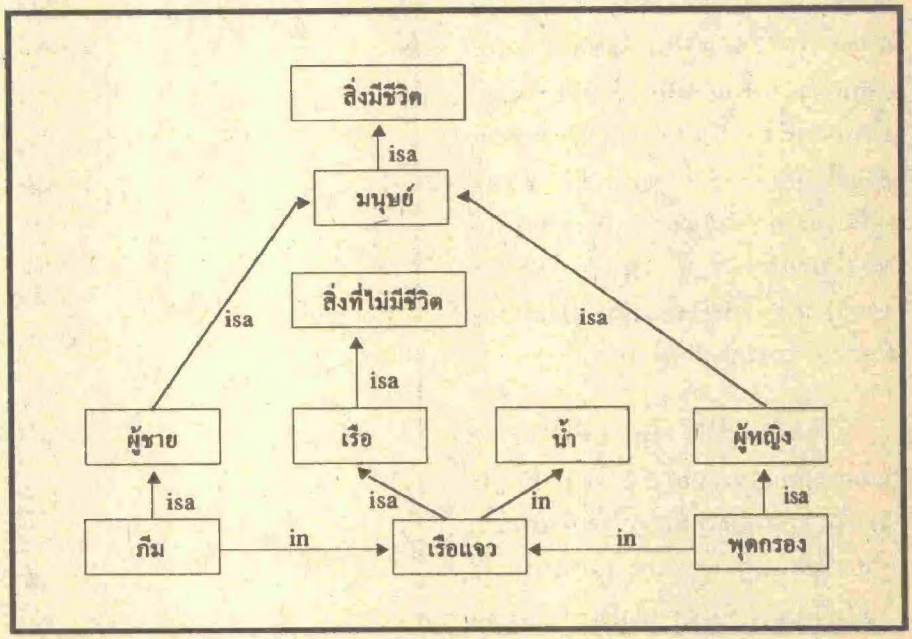
- ข่ายงานแทนความรู้ (semantic networks)
- เฟรมและสคริปต์ (frame and script)
- เพรดิเคตลอจิก (predicate logic)
- กฎของความรู้ (production rules)

ข่ายงานแทนความรู้

รูปแทนความรู้ชนิดข่ายงานจะประกอบขึ้นด้วย

- **โหนด (nodes)** สำหรับแทนออปเจ็กต์ อันได้แก่สรรพสิ่งต่าง ๆ คุณสมบัติหรือลักษณะเฉพาะของสรรพสิ่ง
- **ลิงค์ (links)** สำหรับแทนความสัมพันธ์ระหว่างออปเจ็กต์

ตัวอย่างที่ 1 การแทนประโยค "คุณภีมและพุดกรองพายเรือแจวเล่น" ด้วยข่ายงานแทนความหมายจะมีลักษณะดังรูปที่ 1



รูปที่ 1 ตัวอย่างข่ายงานแทนความหมาย

ข่ายงานแทนความหมายในรูปที่ 1 แสดงให้เห็นถึงความสัมพันธ์ระหว่างข้อมูลในประโยคกับข้อมูลที่สัมพันธ์กัน แต่ไม่ได้กล่าวถึง นั่นก็คือ ภีมเป็นผู้ชาย ผู้ชายเป็นมนุษย์ มนุษย์คือสิ่งมีชีวิต เรือแจวเป็นเรือชนิดหนึ่ง เรืออยู่ในน้ำ พุดกรองและภีมอยู่ในเรือแจว

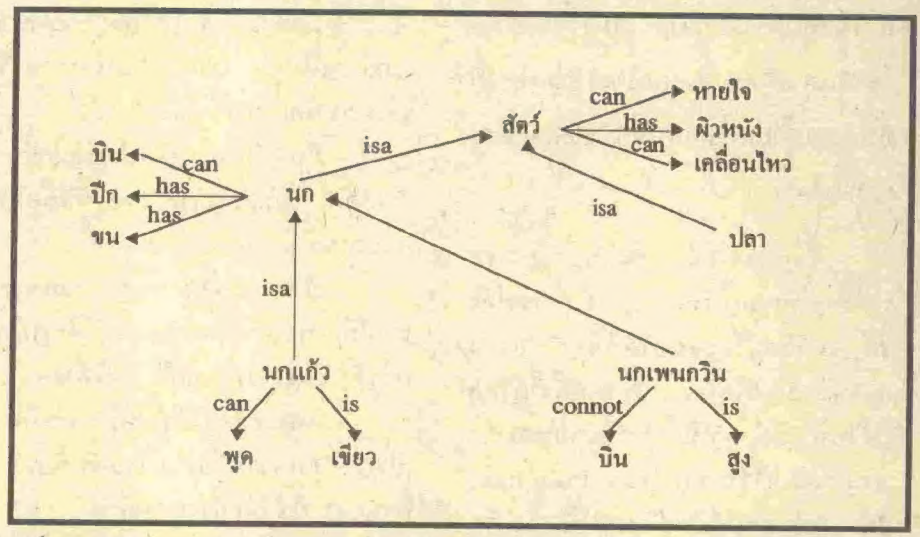
ตัวอย่างที่ 2 การแทนความรู้เกี่ยวกับนกแก้ว ดังแสดงในรูปที่ 2

ตัวอย่างนี้แสดงให้เห็นถึงความสัมพันธ์ระหว่างออปเจ็กต์ ซึ่งแบ่งได้ 2 กลุ่มใหญ่ ๆ คือ

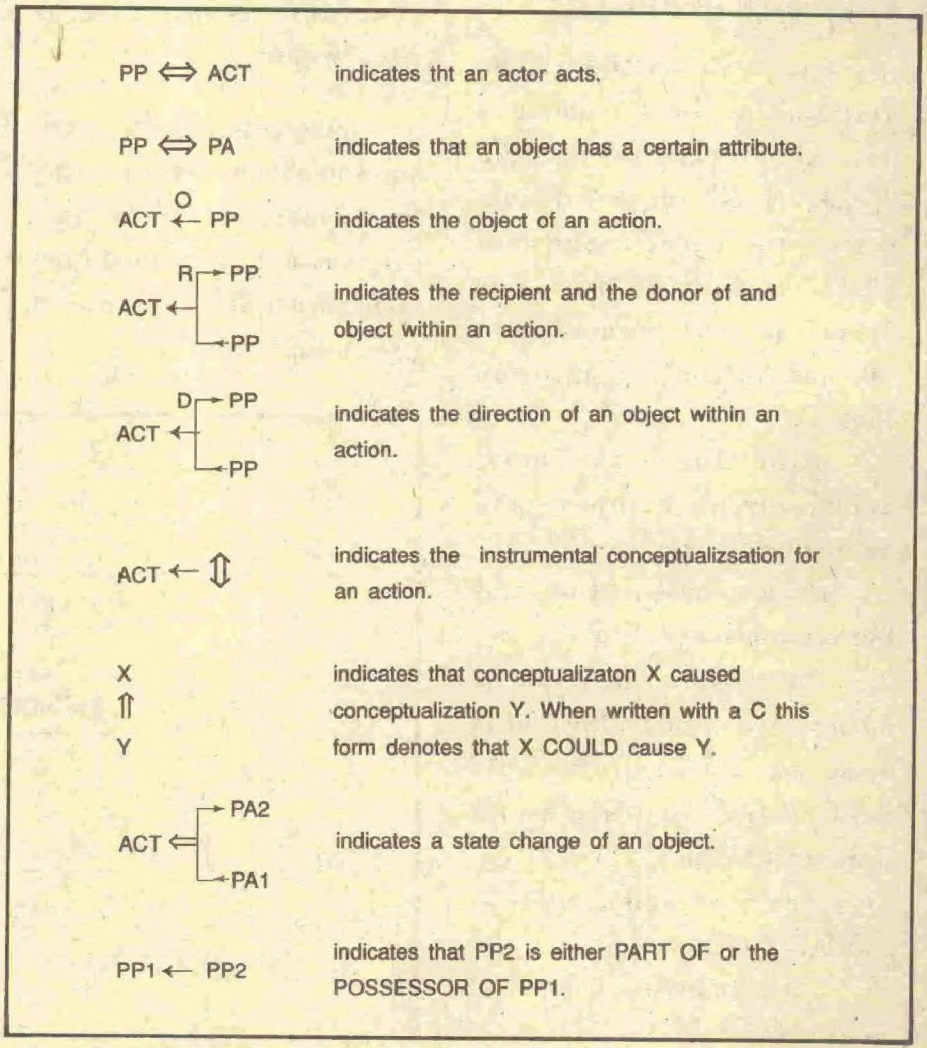
- **คลาส** - ความเป็นสมาชิก (class-membership) ได้แก่ ลิงค์ isa
- **ออปเจ็กต์** - คุณสมบัติของออปเจ็กต์ ได้แก่ ลิงค์ can, cannot, has, is

การแทนความหมายด้วยข่ายงานทำให้คอมพิวเตอร์สามารถถ่ายทอดคุณสมบัติผ่านลิงค์ isa ได้ ตัวอย่างเช่น ถ่ายทอดคุณสมบัติทั่วไปของสัตว์ให้กับนก และของนกให้กับนกแก้ว การแทนความรู้ในลักษณะถ่ายเทคุณสมบัติได้ช่วยทำให้ขนาดของความรู้เล็กลงได้ รูปแทนความรู้ที่จัดอยู่ในกลุ่มของข่ายงานแทนความหมายอีกอันหนึ่งก็คือ ทฤษฎีมโนภาพพึ่งพา (Conceptual dependency theory) (ดูรูปที่ 3) ทฤษฎีการแทนความรู้แบบนี้เป็นความคิดของโรเจอร์ แชงค์ (Roger Shank 1973) ส่วนใหญ่ใช้ในระบบประมวลผลภาษาธรรมชาติ (ดูรูปที่ 4)

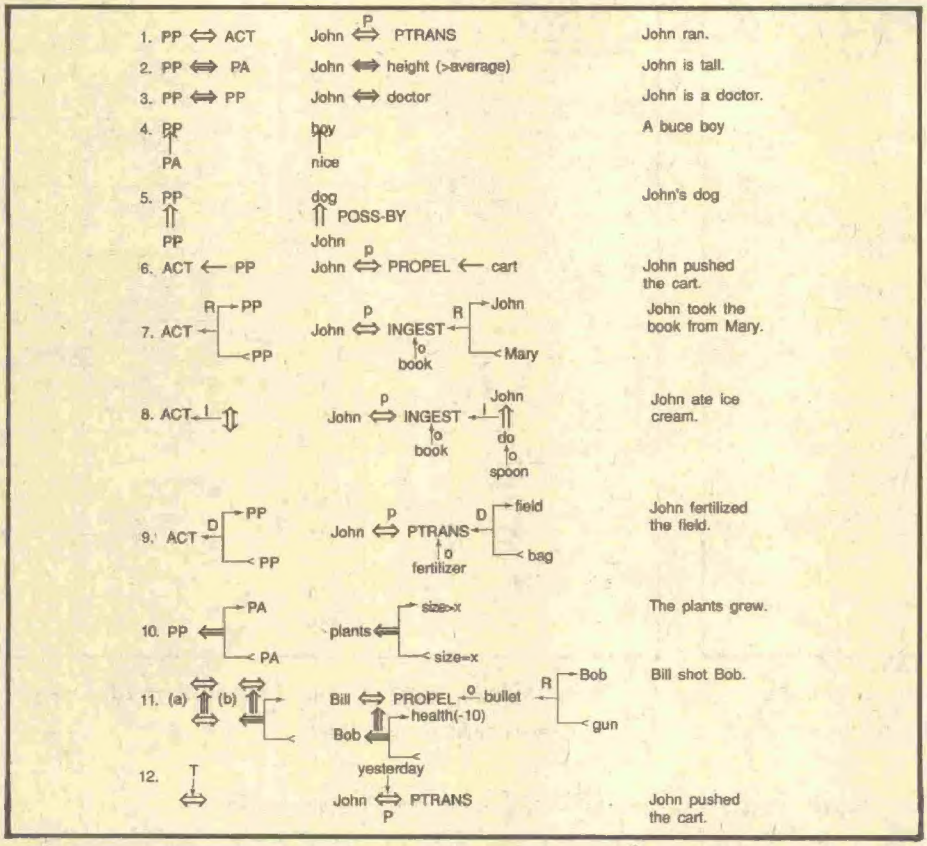
นอกจากนี้ยังมีกราฟมโนภาพ (conceptual graph) ของโซวา (Sowa 1984) ที่เป็นรูปแทนความรู้ที่พัฒนาจากข่ายงานแทนความหมาย กราฟมโนภาพไม่ได้ใช้ลิงค์ที่ถูกกำกับความหมาย (Labeled links) แต่ใช้โหนดเชื่อมความ



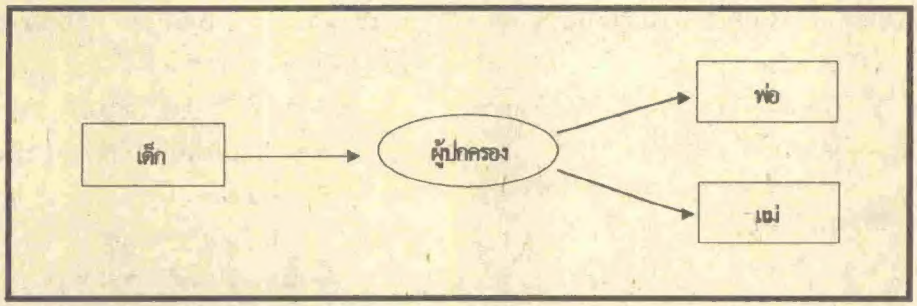
รูปที่ 2 ข่ายงานแทนความหมาย "นกแก้ว"



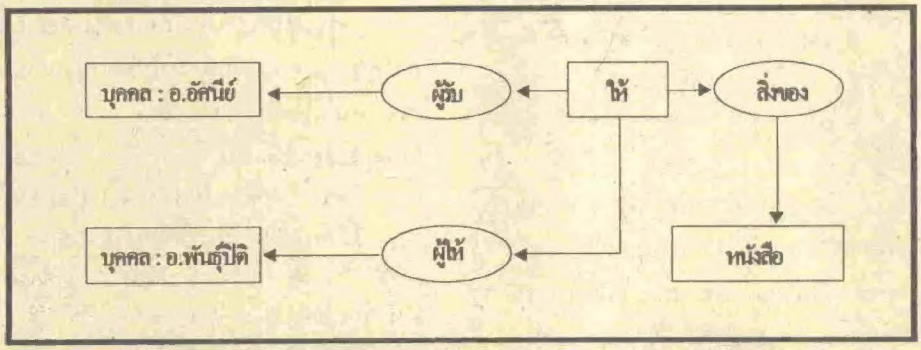
รูปที่ 3 ทฤษฎีมโนภาพพึ่งพา



รูปที่ 4 ตัวอย่างการแทนประโยคภาษาอังกฤษด้วยกราฟมโนภาพพึ่งพา (Rich 1983)



รูปที่ 5 ความสัมพันธ์เชิงมโนภาพ



รูปที่ 6 การแทนประโยค "อ.พันธุ์ปิติให้หนังสือแก่ อ.อัศนีย์ ด้วยกราฟมโนภาพ

สัมพันธ์ทางมโนภาพ ตัวอย่างเช่น ผู้ปกครอง เป็นมโนภาพสัมพันธ์ที่เชื่อมระหว่างมโนภาพเด็กกับพ่อและแม่ (ดูรูปที่ 5)

และในรูปที่ 6 แสดงการใช้กราฟมโนภาพแทนประโยค

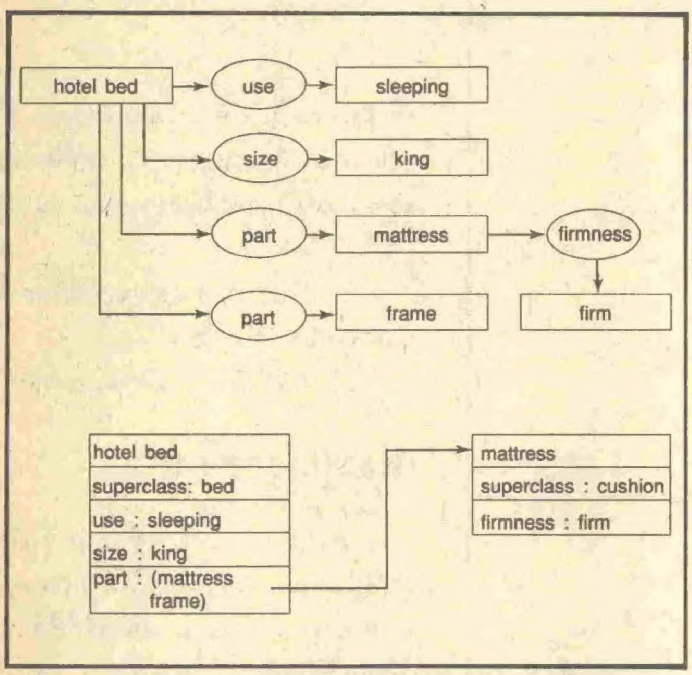
"อ.พันธุ์ปิติให้หนังสือแก่ อ.อัศนีย์"

เฟรมและสคริปต์

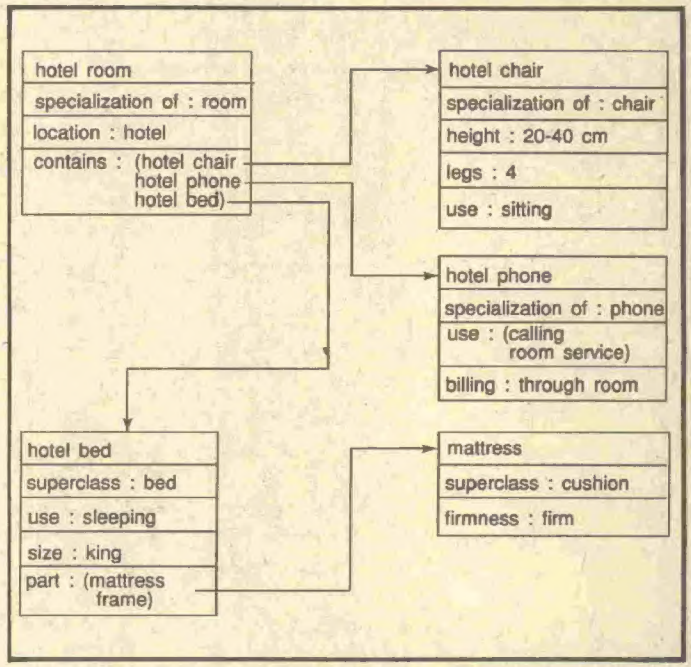
ความรู้ของมนุษย์จะถูกจัดให้สัมพันธ์กันอย่างมีโครงสร้าง ด้วยเหตุนี้จึงมีการประดิษฐ์รูปแทนความรู้ที่เรียกว่าเฟรมและสคริปต์

เราลองมาเปรียบเทียบระหว่างการแทนความรู้เกี่ยวกับ "hotel bed" ด้วยกราฟมโนภาพกับเฟรม (ดูรูปที่ 7)

การแทนความรู้ด้วยเฟรมเป็นแนวคิดของมินสกี (Minsky 1975) เฟรมสามารถจัดความรู้ที่เกี่ยวข้องกันให้เป็นลักษณะโครงสร้าง (structured knowledge) และสามารถจำลองให้เห็นรายละเอียดหรือสถานการณ์โดยรวม (stereotyped situation) เฟรมจะประกอบขึ้นด้วยสล้อต (slots) กับค่าของสล้อต (slot values หรือบางตำราเรียกว่า filler) สล้อตของเฟรมเปรียบได้กับโหนดแทนความสัมพันธ์ระหว่างมโนภาพของกราฟมโนภาพ แต่ค่าของสล้อตนั้นเป็นได้ทั้งค่าที่อิสระ และพอยน์เตอร์หรือดัชนี (pointers) ที่จะชี้ต่อไปยังเฟรมอื่นอีก ดูรูปที่ 7 เราจะเห็นว่าค่าของสล้อตที่ชื่อว่า part จะเป็นพอยน์เตอร์ที่ชี้ไปที่เฟรมของ mattress (ฟูกหรือที่นอน) ในรูปที่ 8 เราจะเห็นได้ว่าคำว่า "hotel bed" นั้นเกี่ยวข้องกับรายละเอียดอื่น ๆ อย่างมีโครงสร้างได้อย่างไร



รูปที่ 7 การแทนความรู้เกี่ยวกับ "hotel bed" ด้วยกราฟมโนภาพและเฟรม (รูปจาก George F Luger และ ผู้ร่วมงาน, 1993)



รูปที่ 8 เฟรมสำหรับแทนความรู้ "hotel room" (ภาพจาก George F Luger และผู้ร่วมงาน, 1993

สคริปต์เป็นอีกตัวอย่างหนึ่งของการแทนความรู้ที่มีลักษณะโครงสร้างและสะท้อนให้เห็นภาพโดยรวมของเหตุการณ์ที่เป็นที่ยอมรับกันโดยทั่วไป ตัวอย่างเช่น เหตุการณ์ที่เกิดขึ้นในภัตตาคารหรือร้านอาหาร มักจะประกอบขึ้นด้วยเรื่องราวของการเข้าไปในร้าน เรื่องราวของการสั่งอาหาร เรื่องราวของการทานอาหาร เรื่องราวของการออกจากร้าน (ดูรายละเอียดเพิ่มเติมจาก ไมโครคอมพิวเตอร์ ฉบับที่ 98)

ทั้งเฟรมและสคริปต์เป็นรูปแทนความรู้ที่จัดความรู้ให้รวมอยู่เป็นกลุ่มเป็นก้อน โดยทั่วไปมักจะแทนความรู้ที่เกี่ยวข้องกับชีวิตประจำวัน

เพรดิเคตลอจิก

ความพยายามที่จะทำให้คอมพิวเตอร์สามารถหาเหตุผลได้อย่างมีจิตใต้สำนึก (commonsense seasoning) เราจำเป็นต้องมีรูปแทนความรู้ที่คำนวณได้ ที่ให้ความหมายไม่กำกวม และพิสูจน์ได้ว่าการอนุมานหาข้อสรุปหรือข้อเท็จจริงใหม่จากความรู้เดิมนั้นถูกต้อง ตัวอย่างเช่น

(1) นิสิตทุกคนทำโครงงานวิศวกรรม (มีความหมายได้ 2 อย่าง)

1.1 นิสิตทุกคนร่วมกันทำโครงงานวิศวกรรมชิ้นเดียว

1.2 นิสิตแต่ละคนต่างทำโครงงานวิศวกรรมของตัวเอง

(2) มนุษย์ทุกคนต้องตาย (ข้อเท็จจริง)

...2.1

แพท นิกสัน เป็นมนุษย์ (ข้อเท็จจริง)

...2.2

แพท นิกสัน ต้องตาย (ข้อสรุปจาก 2.1 และ 2.2)

เพรดิเคตลอจิกเป็นรูปแทนความรู้ที่สามารถแทนความหมายใน (1) ได้อย่างชัดเจน และสามารถคำนวณหาข้อสรุปใน (2) ได้

การแทนประโยคเพรดิเคตลอจิกจะประกอบขึ้นด้วย

- ส่วนที่เป็นเพรดิเคต (predicates) ใช้แทนลักษณะหรือคุณสมบัติของสิ่งต่าง ๆ (objects) หรือความสัมพันธ์ระหว่างสิ่งสองสิ่ง
- ส่วนที่เป็นอาร์กิวเมนต์ (arguements) ใช้แทนชื่อหรือค่าของสิ่งต่าง ๆ ตัวอย่างเช่น

ตาย (แพท___นิกสัน)
เพรดิเคตแทนคุณสมบัติ อาร์กิวเมนต์
ชอบ (ภีม, พุดกรอง)
เพรดิเคตแทนความสัมพันธ์

การแทนความรู้ด้วยเพรดิเคตลอจิกทำให้เราสามารถแทนความรู้ที่เป็นประโยคเชิงซ้อนได้ ตัวอย่างเช่น

จุ๋มและจิ๋มชอบไอศกรีม
= ชอบ (จุ๋ม, ไอศกรีม) ∧ ชอบ (จิ๋ม, ไอศกรีม)
ไม่จุ๋มก็จิ๋มคนใดคนหนึ่งชอบไอศกรีม
= ชอบ (จุ๋ม, ไอศกรีม) ∨ ชอบ (จิ๋ม, ไอศกรีม)
จุ๋มไม่ชอบไอศกรีม
= ~ ชอบ (จุ๋ม, ไอศกรีม)

นอกจากนี้เพรดิเคตลอจิกยังมีสัญลักษณ์บ่งบอกปริมาณ (Quantifiers) เพื่อแทนความหมายที่มีลักษณะทั่วไป คือ ∀ แทน for all, ∃ แทน for some ตัวอย่างเช่น

มนุษย์ทุกคนต้องตาย
∀x : มนุษย์(x)→ตาย(x)
ผู้ชายบางคนเป็นทหาร
∃x : ผู้ชาย(x) ∧ ทหาร(x)

ขอย้อนกลับไปที่ (1), (2) เพรดิเคตลอจิกจึงเป็นรูปแทนความรู้ที่สามารถแทนความหมายที่กำกวมใน (1) ให้ชัดเจนได้ดังนี้

(1.1) ∃x∀y [นักเรียน(y)
∧ โครงงาน(x)-> ทำ(y,x)]
(1.2) ∀y∃x [นักเรียน (y)
∧ โครงงาน(x)-> ทำ(y,x)]

และสามารถคำนวณข้อสรุปในข้อ (2)

(2) ∀x : มนุษย์(x)→ตาย(x)
มนุษย์(แพท___นิกสัน)
∴ ตาย(แพท___นิกสัน)

โดยใช้กระบวนการหาเหตุผลอัตโนมัติที่เรียกว่า เรโซลูชัน (resolution) (รายละเอียดให้ดูเพิ่มจากหนังสืออ้างอิง)

โดยสรุป คุณสมบัติเด่นของเพรดิเคตลอจิกก็คือ เป็นรูปแทนความรู้ที่สามารถแทนกฎเกณฑ์ที่เป็นลักษณะทั่วไปได้ ทำให้คอมพิวเตอร์สามารถหาเหตุผลในเชิงลักษณะได้ (Qualitative reasoning) ดังใน (2)

กฎของความรู้

เพื่อที่จะอธิบายว่ามนุษย์สามารถหาเหตุผลได้อย่างไร นักวิจัย AI ได้สร้างระบบที่เรียกว่าโปดักชันซีสเต็มส์ขึ้นเป็นโมเดลของการประมวลผล ระบบนี้ประกอบขึ้นด้วย 2 ส่วน คือ

● **กฎของความรู้ (production rules)**

ซึ่งอยู่ในรูปของสเตทเมนต์ "If-then" ส่วนของ If จะเรียกว่าส่วนหน้า (antecedent) เป็นส่วนที่บอกถึงเงื่อนไขเหตุการณ์ หรือข้อสมมติฐาน และส่วนของ then จะเรียกว่าส่วนตาม (consequent) ซึ่งเป็นส่วนของการกระทำหรือบทสรุป

● **ส่วนทำงาน (working memory)**

คือส่วนที่ประมวลผลข้อเท็จจริง (fact) กับกฎเพื่อหาข้อเท็จจริงใหม่

ลองดูตัวอย่างต่อไปนี้

"ภุชงค์ตื่นขึ้นมาและมองไปที่ปฏิทิน วันนี้เป็นวันที่ 3 มกรา" "วันที่ 3 มกรา" เข้าไปในส่วนทำงาน และดึงกฎที่ว่า "If วันนี้เป็นวันที่ 3 มกรา then วันนี้เป็นวันเกิดของเพื่อนหญิงของเขา" ภุชงค์จะนึกได้ทันทีว่า "วันนี้เป็นวันเกิดของสมศรี" และถ้าหากภุชงค์ยังคงคิดต่อกฎอื่นก็จะตามมา เช่น "If วันเกิด then ควรซื้อของขวัญ" ฉะนั้นภุชงค์ก็จะได้ความคิดใหม่ต่อไปว่า "เขาควรซื้อของขวัญให้สมศรี"

การหาเหตุผลโดยอัตโนมัติ

เมื่อมีรูปแทนความรู้ที่สามารถคำนวณได้ การหาเหตุผล การตัดสินใจ การสร้างความรู้ขึ้นใหม่ หรือหาข้อสรุปโดยเครื่องคอมพิวเตอร์ก็กระทำได้ง่ายขึ้น กลไกในการหาเหตุผลที่สำคัญมีอยู่ 2 แบบ ดังนี้

1. การหาเหตุผลแบบอนุกรมเดินหน้า (forward chaining)
2. การหาเหตุผลแบบอนุกรมถอยหลัง (backward chaining)

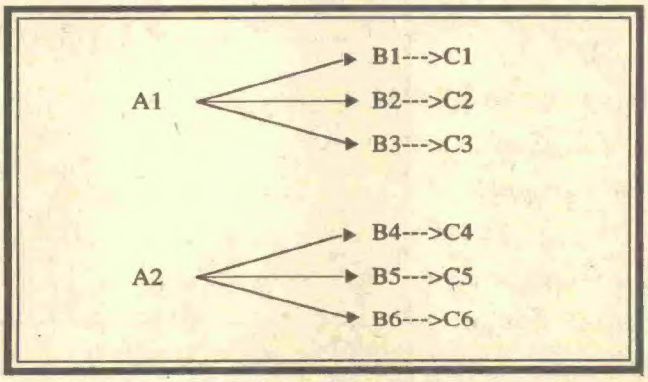
สมมติว่าเรามีข้อเท็จจริง กฎ และเป้าหมาย ดังต่อไปนี้

ข้อเท็จจริง : A1, A2
กฎ : A1 ---> B1, A1 ---> B2, A1 ---> B3
A2 ---> B4, A2 ---> B5, A1 ---> B6
B1 ---> C1, B2 ---> C2, B3 ---> C3
B4 ---> C4, B5 ---> C5, B6 ---> C6
เป้าหมาย : C3

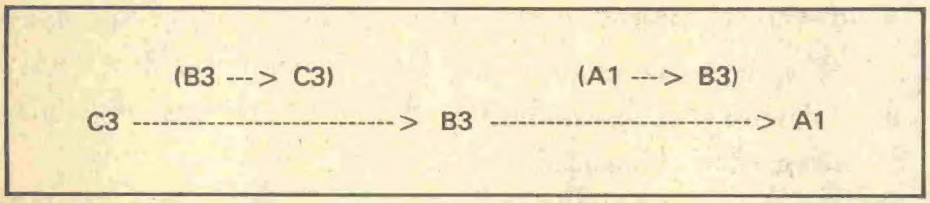
หมายเหตุ
ด้านหางลูกศรคือส่วนของ if และด้านหัวลูกศรคือส่วนของ then

การหาเป้าหมายหรือคำตอบ C3 โดยใช้การหาเหตุผลแบบอนุกรมเดินหน้า กล่าวคือ เริ่มต้นจากข้อเท็จจริงที่มีอยู่ จะมีลักษณะดังรูปที่ 9

แต่การใช้การหาเหตุผลแบบอนุกรมถอยหลัง จะเริ่มต้นจากเป้าหมายแล้วตรวจสอบดูว่ามีข้อเท็จจริง A1 หรือ A2 หรือไม่ การหาเหตุผลแบบอนุกรมถอยหลังจะมีลักษณะดังรูปที่ 10



รูปที่ 9
การหาเหตุผลแบบอนุกรมเดินหน้า



รูปที่ 10 การหาเหตุผลแบบอนุกรมถอยหลัง

การแทนความรู้ด้วยกฎและการหาเหตุผลทั้ง 2 แบบ จะนำไปประยุกต์ใช้กันมากในระบบผู้เชี่ยวชาญ แต่เนื่องจากระบบผู้เชี่ยวชาญต้องประกอบด้วยกฎมากมายนับเป็นจำนวนพัน ฉะนั้นจึงต้องมีวิธีการลดจำนวนการค้นหากฎ ซึ่งเราจะยังไม่พูดถึงในที่นี้

การจัดการความรู้หรือการหาเหตุผลโดยเครื่องจักร ยังต้องคำนึงถึงชนิดของข้อมูล ซึ่งแบ่งออกเป็น 2 กลุ่ม คือ

1. กลุ่มข้อมูลที่ชัดเจน คือข้อมูลที่มีเส้นแบ่งระหว่างเท็จหรือจริง แน่นอนหรือไม่แน่นอน ชัดเจนหรือไม่ชัดเจนอย่างเห็นได้ชัด

ตัวอย่างเช่น 4 มีค่ามากกว่า 3, if เด็กสูงกว่า 190 เซนติเมตร then เด็กต้องจ่ายค่าโดยสารครึ่งราคา

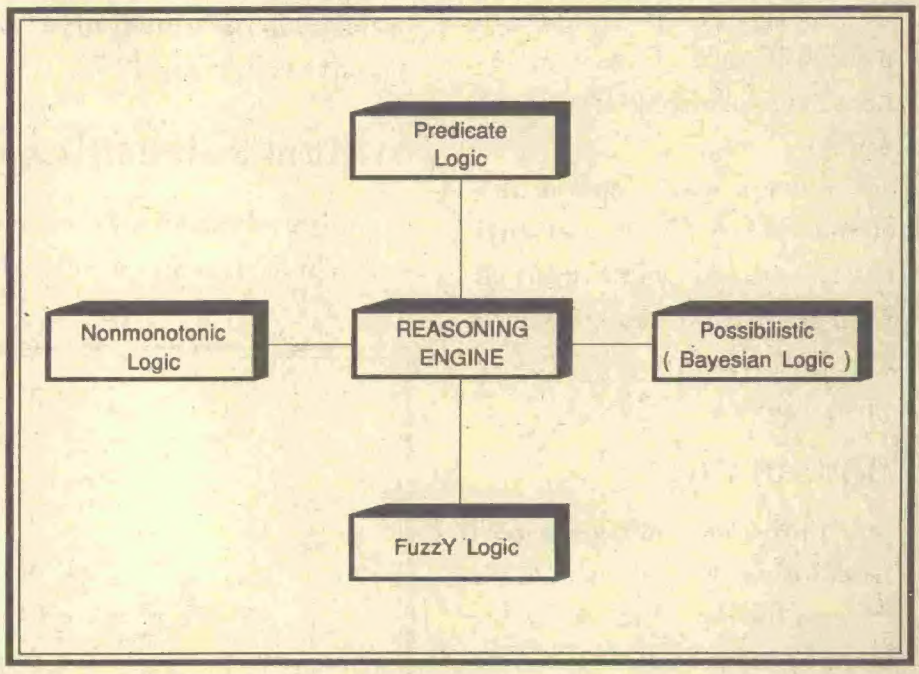
2. กลุ่มข้อมูลที่ไม่ชัดเจน คือข้อมูลที่ไม่สามารถให้ข้อสรุปที่แน่นอน ข้อมูลที่เกี่ยวกับความเชื่อ

ตัวอย่างเช่น if จำนวนขายสูง และจำนวนสินค้าในสต็อคต่ำ then ให้เพิ่มปริมาณการสั่งสินค้า

ด้วยเหตุนี้จึงมีการพัฒนารูปแทนความรู้จากเพรดิเคตลอจิกมาเป็นฟัซซี่ลอจิก (Fuzzy Logic) ที่มีดีกรีความไม่ชัดเจน และเบย์เซียนลอจิก (Bayesian Logic) ที่มีดีกรีความน่าจะเป็น ส่วนการหาเหตุผลจากกฎก็มีการพัฒนาจากลักษณะที่เคยให้ความมั่นใจในข้อสรุปร้อยเปอร์เซ็นต์ มาเป็นข้อสรุปที่กำหนดดีกรีของความแน่นอนหรือความมั่นใจ (CF or Certainty Factors) และมาเป็นข้อสรุปที่สามารถปรับเปลี่ยนได้โดยใช้นอนโมโนโทนิคลอจิก (Non-monotonic Logic หรือ Non-deterministic Logic) เทคนิคในการหาเหตุผลโดยเครื่องจักร สามารถสรุปได้ดังรูปที่ 11

เครื่องมือสำหรับพัฒนาความรู้

การประดิษฐ์ความรู้ให้กับคอมพิวเตอร์จำเป็นต้องมีภาษาโปรแกรมและเครื่องมือช่วยในการพัฒนาความรู้



รูปที่ 11 การหาเหตุผลโดยเครื่องจักร

ภาษาโปรแกรมที่นิยมใช้มีอยู่ 2 ภาษา คือ ลิสป์ (LISP) และโปรล็อค (PROLOG)

LISP

ลิสป์เป็นภาษาระดับสูงเช่นเดียวกับภาษาฟอร์แทรนหรือโคบอล จอห์น แมคคาธีและเพื่อนร่วมงานของเขาเป็นผู้พัฒนาซอฟต์แวร์นี้ขึ้นมา ลักษณะโปรแกรมที่เขียนด้วยภาษาลิสป์สามารถสังเกตเห็นได้ง่าย กล่าวคือ แต่ละบรรทัดมักจะขึ้นต้นด้วยวงเล็บเปิด "("

ภาษานี้เหมาะสำหรับแสดงและจัดการเกี่ยวกับมโนภาพ ซึ่งก็คือการประมวลผลเชิงสัญลักษณ์ (Symbolic processing)

ตัวอย่าง เราสามารถแสดงความหมายหรือมโนภาพของ ARC ในเรื่องของ BLOCK ได้ดังนี้

```
(ARC (PARTS X Y Z)
     (X IS-A BLOCK)
     (Y IS-A BLOCK)
     (Z IS-A BLOCK)
     (X HAS-ORIENTATION STANDING)
     (Y HAS-ORIENTATION STANDING)
     (Z HAS-ORIENTATION LYING)
     (X SUPPORTS Z)
     (Y SUPPORTS Z)
     (X MUST-NOT-TOUCH Y)
     (Y MUST-NOT-TOUCH X)          )
```

นอกจากนี้เรายังสามารถแทนฐานความรู้ด้วยภาษาลิสป์ได้เช่นกัน

ตัวอย่าง การแทนความรู้เกี่ยวกับมหาวิทยาลัยเกษตรศาสตร์

```
(KU (A-KIND-OF UNIVERSITY)
    (FULLNAME KASETSART-UNIVERSITY)
    (LOCATION PAHOLYOTHIN-Rd-THAI-
    LAND)
    (PHONE 5790113)                )
```

และในระบบผู้เชี่ยวชาญก็นิยมใช้ภาษาลิสป์ในการเขียนกฎ ในโครงสร้างเดียวกันกับตัวอย่างข้างต้น

ตัวอย่าง กฎสำหรับอนุมานว่า x เป็นนก ถ้า x มีปีก

```
(RULE12
        (IF (HAS-A WING)
        (THEN (IS-A BIRD)))
```

วัตถุประสงค์หลักของการใช้ภาษาลิสป์คือเพื่อการประมวลผลนิพจน์เชิงสัญลักษณ์ ไม่ใช่การคำนวณตัวเลข เนื่องจากในโปรแกรมภาษาลิสป์ไม่มีความแตกต่างระหว่างตัวโปรแกรมกับข้อมูล (data) ฉะนั้นโปรแกรมลิสป์จะสามารถพิจารณาโปรแกรมอื่นหรือแม้กระทั่งตัวมันเองว่าเป็นนิพจน์เชิงสัญลักษณ์ และจากนั้นก็ทำการประเมินหาคำตอบ

ภาษาลิสป์ยังแตกแขนงออกไปเป็น INTERLISP, MACLISP, FRANZLISP, ZETALISP ทำให้โปรแกรมลิสป์ไม่สามารถย้ายไปมา (portable) ได้สะดวก อย่างไรก็ดี ยังมี COMMONLISP ที่ได้รับการยอมรับว่าเป็นภาษามาตรฐานและเป็นที่นิยมใช้กันอย่างกว้างขวาง

PROLOG

โปรล็อกเป็นภาษาที่ได้รับการพัฒนาจาก Colmerauer และ Kowalski และเป็นที่นิยมใช้ในค่าย AI ของยุโรปและญี่ปุ่น โปรล็อกได้รับเลือกให้เป็นภาษาหลักในโครงการการพัฒนาคอมพิวเตอร์ยุคที่ 5 ของประเทศญี่ปุ่น (Japan's Fifth Generation Computer System Project) รายละเอียดความรู้เบื้องต้นเกี่ยวกับโปรล็อกหาอ่านได้จากหนังสือของ Clocksin และ Mellish (1981)

ตัวอย่างในการใช้ภาษาโปรล็อกเพื่อแสดงข้อเท็จจริงว่า "จอห์นเป็นพ่อของแมรี่" เขียนได้ดังนี้

```
father(John, Mary)
```

และเมื่อจะตั้งคำถามว่า "ใครเป็นพ่อของแมรี่" เราสามารถใช้ภาษาโปรล็อกแสดงแทนได้ดังนี้

```
? - father(X, Mary)
```

จะได้รับตอบว่า

```
X = John
```

โปรล็อกยังสามารถแสดงแทนประโยคเงื่อนไขหรือกฎได้ดังนี้

```
"X เป็นผู้ปกครองของ Y ถ้า X เป็นพ่อ
ของ Y"
```

เมื่อแสดงอยู่ในรูปของโปรล็อกจะได้

```
parent(X, Y) :- father(X, Y)
```

ฉะนั้นการใช้ภาษาโปรล็อกจะช่วยให้ตอบคำถามที่ซับซ้อนขึ้นได้ ภาษาโปรล็อกเหมาะสำหรับปัญหาที่สามารถอธิบายได้ด้วยนิพจน์ตรรก ดังตัวอย่างข้างต้น กล่าวคือ อธิบายด้วยข้อเท็จจริงและเงื่อนไข การหาคำตอบหรือการแก้ปัญหาของภาษาโปรล็อกก็คือการพิสูจน์ค่าความเป็นจริงของข้อเท็จจริง และการผูกตัวแปร (binding) ด้วยวิธียูนิฟิเคชัน (unification)

อย่างไรก็ดี ได้มีคนพยายามที่จะรวมเอาข้อดีของลิสป์และโปรล็อกมาไว้อยู่ในภาษาเดียวกัน ตัวอย่างเช่น

```
LOGLISP, POPLOG
```

นอกจากนี้ยังมีการเตรียมภาษาโปรแกรมที่เรียกว่า Object-oriented language ภาษานี้เราจะเรียกว่า OOPS (Object-Oriented Programming Language) ลักษณะเด่นของภาษานี้คือสนับสนุนการถ่ายเทคุณสมบัติระหว่างออปเจ็กต์ที่อยู่ในคลาสเดียวกัน การเขียนโปรแกรมเชิงออปเจ็กต์ก็คือ การกำหนดให้ข้อมูลอยู่ในลักษณะของออปเจ็กต์ โดยโครงสร้างของโปรแกรมจะเกิดจากการเชื่อมโยงออปเจ็กต์ต่าง ๆ เข้าด้วยกันดังรูปที่ 12

OOPS

ฟังก์ชันที่สำคัญใน OOPS คือ ฟังก์ชันในการกำหนดออปเจ็กต์ขึ้นใหม่ (def-object) และให้อยู่ในลักษณะลำดับชั้น (hierarchy), ฟังก์ชันในการเชื่อมกระบวนการคำนวณ (procedures) ให้กับออปเจ็กต์ (def-method) และฟังก์ชันสำหรับส่งผ่านข้อมูลระหว่างออปเจ็กต์ (def-message) ตัวอย่างเช่น

1) กำหนดให้ออปเจ็กต์มีโครงสร้างเป็นลำดับขั้น

```
(def-object 'rectangle 'object
'((numsides 4) (description "four-sided
figure, all angles equal 90°")))
```

ในที่นี้ def-object เป็นฟังก์ชันที่กำหนดให้สี่เหลี่ยมผืนผ้าอยู่ในคลาสของออปเจ็กต์ โดยมีคุณสมบัติพิเศษคือประกอบด้วยด้าน 4 ด้าน แต่ละด้านประกบกันเป็นมุม 90 องศา

ฟังก์ชัน def-object ประกอบด้วยพารามิเตอร์ 3 ตัว พารามิเตอร์แรกคือออปเจ็กต์ที่กำหนดขึ้นใหม่ พารามิเตอร์ตัวที่สองคือ ออปเจ็กต์เป็นคลาสของพารามิเตอร์ตัวแรก และพารามิเตอร์ตัวสุดท้ายคือคุณสมบัติของพารามิเตอร์ตัวแรก

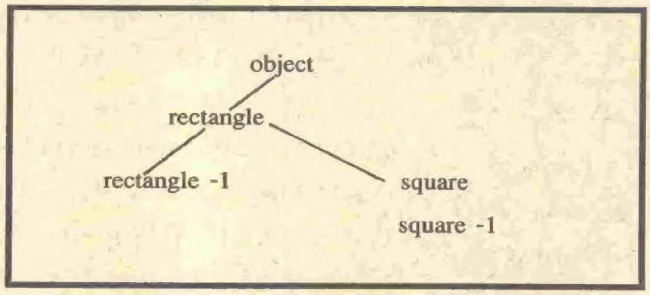
ดังนั้นฐานความรู้ในรูปที่ 12 เมื่อแทนด้วย OOPS จะมีลักษณะดังต่อไปนี้

```
(def-object 'rectangle-1 'rectangle
'((length 8) (width 4)))
(def-object 'square 'rectangle
'((description "rectangle whose sides are
equal")))
(def-object 'square-1 'square '((side 10)))
```

2) กำหนดกระบวนการคำนวณให้กับออปเจ็กต์

```
(def-method 'rectangle 'area
'(lambda( ) (*length width)))
```

ในที่นี้ def-method กำหนดวิธีคำนวณพื้นที่ให้กับกลุ่มออปเจ็กต์ในคลาส rectangle def-method ใช้พารามิเตอร์ 3 ตัวเช่นกัน ตัวแรกเป็นชื่อของออปเจ็กต์ที่ต้องการกำหนดกระบวนการคำนวณให้ ตัวที่ 2 คือชื่อของกระบวนการคำนวณ ตัวที่ 3 คือวิธีการคำนวณหรือขั้นตอนการคำนวณ กระบวนการ "area" สามารถถ่ายทอดให้กับออปเจ็กต์ที่อยู่ในคลาส "rectangle" แต่บางกรณีต้องมีการยกเว้น เช่น การคำนวณค่าพื้นที่ของสี่เหลี่ยมจตุรัส ในกรณีนี้ต้องกำหนดวิธีคำนวณใหม่ นั่นคือ



รูปที่ 12
ลำดับชั้นของข้อมูลใน OOPS

```
(def-method 'square 'area
'(lambda( ) (expt side 2)))
```

3) การส่งผ่านข้อมูลระหว่างออปเจ็กต์

```
> (message 'rectangle-1 'area)
32
> (message 'square-1 'show-value 'numsides)
4
```

ฟังก์ชันส่งผ่านข้อมูลคือ message ประกอบด้วยพารามิเตอร์ 2 ตัว ตัวแรกคือชื่อออปเจ็กต์ที่จะรับข้อมูล ตัวที่สองคือชื่อกระบวนการคำนวณหรือคุณสมบัติที่ต้องการให้ถ่ายทอดจากออปเจ็กต์ที่อยู่ในคลาสที่เหนือกว่า

นอกจากภาษาโปรแกรมที่กล่าวมาแล้ว ยังมีภาษาสำหรับแทนความรู้โดยตรง เช่น KRL, FRL, KL-ONE และ CYCL เป็นต้น

เครื่องจักรที่เตรียมไว้สำหรับงานปัญญาประดิษฐ์

ไม่เพียงแต่จะมีการพัฒนาภาษาโปรแกรมเพื่องาน AI แต่ยังมีการพัฒนาทางด้านฮาร์ดแวร์ด้วย ได้แก่ เครื่องจักรลิสป์ซึ่งถูกพัฒนาจากหลายบริษัท เช่น LISP Machined Inc., Xerox, Texas Instruments เป็นต้น เครื่องจักรลิสป์พัฒนาขึ้นมาเพื่อสนับสนุนการทำงานของโปรแกรมลิสป์ให้เร็วขึ้น เครื่องจักรเหล่านี้บางเครื่องได้ออกแบบเพื่อสนับสนุนการทำงานของโปรแกรมโปรลือกด้วย

ส่งท้าย

การประดิษฐ์ความรู้ให้กับคอมพิวเตอร์เป็นเพียงจุดเริ่มต้นของการสร้างเครื่องจักรที่คิดได้ หนทางการพัฒนายังมีอีกยาวไกล อย่างไรก็ตาม ผลงานของปัญญาประดิษฐ์ที่ผ่านมาช่วยให้เรามีเครื่องใช้ไม้สอยที่มีประสิทธิภาพมากขึ้น เช่น เครื่องซักผ้าระบบฟัซซี่ที่สามารถปรับการทำงานให้เข้ากับชนิดของผ้า ตลอดจนมีซอฟต์แวร์ที่ช่วยในการตัดสินใจ เช่น TECKCHEK โปรแกรมสำหรับคัดเลือกผู้มาสมัครงาน

หนังสืออ้างอิง

1. Maureen Caudill, Charles Butter, "Naturally Intelligent Systems", MIT press, 1990

2. George F. Luger, William A. Stubbleffield, "Artificial Intelligence : Structures and Strategies for Complex Problem Solving", 2 nd edition, Addison-Wesley, 1993

3. Harvey Bookman, "TECKCHEK : An Expert System that Assesses Expertise", PC AI Jan/Feb. 1993, pp 22-24